本ドキュメントはCypress (サイプレス) 製品に関する情報が記載されております。本ドキュメントには、仕様の開発元企業として「スパンション」，「Spansion」，「富士通」または「Fujitsu」の名が記載されておりますが、これらの製品は Cypress が新規および既存のお客様に引き続き提供してまいります。

商品仕様の継続性について
Cypress 製品として提供することに伴う商品仕様としての変更はなく、ドキュメントとしての変更もありません。また本ページのお知らせは、変更情報として追記いたしません。本ドキュメントに変更情報が記載されている場合、それは本お知らせを除いた前版からの変更点です。なお、今後改訂は必要に応じて行われますが、その際の変更内容は改訂後のドキュメントに記載いたします。

オーダ型格および品名について
Cypress は既存のオーダ型格および品名を引き続きサポートいたします。これらの製品をご注文の際は、このドキュメントに記載されているオーダ型格および品名をご使用ください。

詳しいお問い合わせ先
Cypress 製品およびそのソリューションの詳細につきましては、お近くの営業所へお問い合わせください。

サイプレスについて
サイプレス (銘柄コード：CY) は、車載や産業機器、ネットワーキング プラットフォームから高機能民生機器およびモバイル機器まで、今日の最先端組み込みシステム向けに高性能で高品質のソリューションを提供します。NOR フラッシュ メモリや F-RAM™、SRAM、Traveo™ マイクロコントローラー、業界唯一の PSoC®プログラマブル システムオンチップ ソリューション、アナログおよび PMIC Power Management IC、CapSense®静電容量タッチセンシング コントローラー、Wireless BLE Bluetooth® Low-Energy、USB コネクティビティ ソリューションなど、幅広い差別化製品ポートフォリオを、一貫した革新性と業界最高クラスの技術サポート、比類のないシステム バリューとともにグローバルに提供します。

FUJITSU

AN07-00189-1

F^{2}MC 8FX Family
8-BIT MICROCONTROLLER
MB95200H/210H series

DBG 端子の使い方

注意事項

- 本資料の記載内容は、予告なしに変更することがありますので、ご用命の際は営業部門にご確認ください。
- 本資料に記載された動作概要や応用回路例は、半導体デバイスの標準的な動作や使い方を示したもので、実際に使用する機器での動作を保証するものではありません。したがいまして、これらを使用するにあたってはお客様の責任において機器の設計を行ってください。これらの使用に起因する損害などについては、当社はその責任を負いません。
- 本資料に記載された動作概要・回路図を含む技術情報は、当社もしくは第三者の特許権、著作権等の知的財産権やその他の権利の使用権または実施権の許諾を意味するものではありません。また、これらの使用について、第三者の知的財産権やその他の権利の実施ができることの保証を行うものではありません。したがって、これらの使用に起因する第三者の知的財産権やその他の権利の侵害について、当社はその責任を負いません。
- 本資料に記載された製品は、通常の産業用、一般事務用、パーソナル用、家庭用などの一般的用途に使用されることを意図して設計・製造されています。極めて高度な安全性が要求され、仮に当該安全性が確保されない場合、社会的に重大な影響を与えかつ直接生命・身体に対する重大な危険性を伴う用途（原子力施設における核反応制御、航空機自動飛行制御、航空交通管制、大量輸送システムにおける運行制御、生命維持のための医療機器、兵器システムにおけるミサイル発射制御をいう）、ならびに極めて高い信頼性が要求される用途（海底中継器、宇宙衛星をいう）に使用されるよう設計・製造されたものではありません。したがって、これらの用途にご使用をお考えのお客様は、必ず事前に営業部門までご相談ください。ご相談なく使用されたことにより発生した損害などについては、責任を負いかねますのでご了承ください。
- 半導体デバイスはある確率で故障が発生します。当社半導体デバイスが故障しても、結果的に人身事故、火災事故、社会的な損害を生じさせないよう、お客様は、装置の冗長設計、延焼対策設計、過電流防止対策設計、誤動作防止設計などの安全設計をお願いします。
- 本資料に記載された製品を輸出または提供する場合は、外国為替及び外国貿易法および米国輸出管理関連法規等の規制をご確認の上、必要な手続きをおとりください。
- 本書に記載されている社名および製品名などの固有名詞は、各社の商標または登録商標です。

Copyright© 2008 FUJITSU MICROELECTRONICS LIMITED all rights reserved

改版履歴

版数	日付	内容
1.0 版	2008/10/08	新規作成

目次

注意事項 1
改版履歴 2
目次 3
1 はじめに 4
2 MCUの動作モード 5
2.1 オンチップデバッグシステム 5
2.2 MCUの動作モード 5
2.3 動作モードの切り替え 5
3 DBG 端子の機能 6
3.1 DBG 端子のハードウェア接続 (デバッグモード) 6
3.2 デバッグモードの動作 6
3.3 フリーランモードの動作 6
4 DBG 端子の使用方法 7
4.1 1 ライン UART 端子 / P12 端子としての使用 7
4.2 EC0 端子としての使用 7
5 DBG 端子の使用上の注意事項 9
6 サンプルコード 10

1　はじめに

このアプリケーションノートでは, DBG 端子の使用方法を説明します。DBG 端子は，デバッグモードでは 1 ライン UART 通信ポートとして，フリーランモードでは P12 端子または EC0 端子として使用できます。

2　MCU の動作モード

MCU の動作モードとオンチップデバッグシステムについて

2.1　オンチップデバッグシステム

MB95200 シリーズ MCU は、オンチップデバッグ(OCD)システムを搭載しています。OCD システムは MCU の DBG 端子と専用 BGM アダプタ(MB2146-08-E)を介してデバッガホスト（PC）と通信し、各種デバッグ機能を実現します。

2.2　MCU の動作モード

OCD システムには２つの動作モードがあり、MCU は BGM アダプタとデバッガホスト（PC）との接続の有無によって、それぞれ次のモードで動作します。

・ デバッグモード

デバッガ機能が有効なモードです。MCU は BGM アダプタと通信を行い、各種デバッグ機能が利用可能となります。BGM アダプタとデバッガホスト(PC)が接続されている状態で、MCU はデバッグモードになります。

・ フリーランモード

デバッガ機能が無効なモードです。MCU はユーザコードを実行します。BGM アダプタとデバッガホスト(PC)が接続されていない状態では、MCU はフリーランモードになります。

2.3　動作モードの切り替え

MCU は，パワーオンリセット後の DBG 端子の状態で，動作モードを切り替えます。このため MCU の動作モードを切り替えるには外部リセットではなく，MCU の電源を一旦切断した後，再度入れ直す必要があります。動作モードの切り替えのタイミングについては，「MB95200 シリーズ BGM アダプタ MB2146-08-E 取扱説明書」の 6.章 デバッグモード開始仕様を参照願います。

3 DBG 端子の機能

各モードでの DBG 端子の機能

3.1 DBG 端子のハードウェア接続 (デバッグモード)

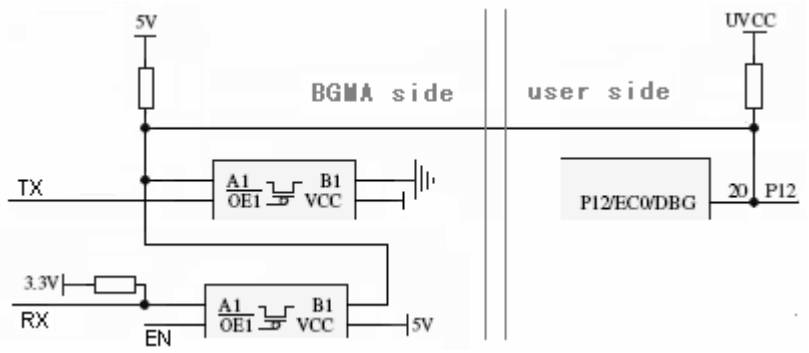

図 2-1 デバッグモードでの DBG 端子のハードウェア接続

3.2 デバッグモードの動作

MCU がデバッグモードに入った場合, DBG 端子は 1 ライン UART 端子として機能し, P12/EC0 端子としての機能は無効になります。このとき, ユーザコード内に P12 端子の処理があっても機能しません。EC0 端子として DBG 端子の代わりに P04 端子を使用できます。SYSC [EC0SL] レジスタに「1」を書き込むと, P04 端子は EC0 端子として機能します。このとき P04 端子の EC0 入力と HCLK1 入力は, 同時には使用できません。

3.3 フリーランモードの動作

MCU がフリーランモードに入った場合, 1 ライン UART の機能は無効になります。P12/EC0 端子としての機能をユーザコード内で使用できます。しかし, ユーザコード内で DBG 端子を 1 ライン UART 端子として設定はできません。

P12/DBG/EC0 端子はオープンドレイン端子です。このため外付けプルアップ抵抗を接続してください。(図 2-1 user side 参照)

4 DBG 端子の使用方法

各モードでの DBG 端子設定の例

4.1 1 ライン UART 端子 / P12 端子としての使用

デバッグモードでは, DBG 端子は 1 ライン UART 端子として使用されます。DBG 端子は UART がアイドル状態のときは常に"H"です。フリーランモードでは,デフォルト設定では P12 として使用されます。ユーザコードを次のように作成する場合を考えます。

```
DDR1_P12 = 1;     // set P12 as output pin
PDR1_P12 = 0;     // pull P12 to low.
```

デバッグモードでこれらのコードを実行しても, DBG 端子は"H"レベルのままです。しかし, フリーランモードでこれらのコードを実行すると, DBG 端子は"L"レベルになります。

4.2 EC0 端子としての使用

デバッグモードでは P12(DBG 兼用) 端子を,EC0 端子(8/16 ビット複合タイマ クロック入力端子)として使用できません。デバッグモードで, 8/16 ビット複合タイマに外部クロックを入力するには, SYSC レジスタの bit3 EC0SL (クロック選択ビット)に"1"を設定し、クロック入力端子の設定を P12 端子から P04 端子に変更します。(初期設定では P12 端子が EC0 入力端子設定となっています)

```
SYSC = 0xEB;      // set P04 as EC0 pin,
T00CR0 = 0x71;    // interval timer, external clock source.
TMCR0 = 0x43;     // 8 bit mode, output T00.
T00DR = 0x08;           // set counter.
T00CR1 = 0x81;    // enable output, start T00.
```

上記のサンプルコードは, P04 端子を EC0 端子として設定し, タイマ 00 を連続インターバルタイマモードで設定します。クロック信号を P04 端子に入力し, これらのコードをデバッグモードで実行し, T00 端子をチェックすると, 出力クロックが入力クロック / 8 であることがわかります。

```
SYSC = 0xE3;      // set P12 as EC0 pin,
T00CR0 = 0x71;    // interval timer, external clock source.
TMCR0 = 0x43;     // 8bit mode, output T00.
T00DR = 0x08;           // set counter.
T00CR1 = 0x81;    // enable output, start T00.
```

上記のサンプルコードは，フリーランモードにおいて、P12(DBG 兼用)端子 を EC0 端子として利用するサンプルコードです。SYSC レジスタの bit3 EC0SL (クロック選択ビット)に”0”を設定し、EC0 端子をクロック入力端子に設定しています。 タイマ 00 を連続インターバルタイマモードで設定します。クロック信号を P12 端子に入力し，これらのコードをフリーランモードで実行し，T00 端子をチェックすると，出力クロックが入力クロック / 8 であることがわかります。

5 DBG 端子の使用上の注意事項

1. BGM アダプタを使用してデバッグを行う場合, DBG 端子に接続された周辺回路が 1 ライン UART 通信に影響しないように設計する必要があります。ユーザボードを設計するときは, 図 2-1 を参照してください。

2. デバッグを行うときには, ユーザコード内で P12(DBG 兼用)端子を出力"H"設定ではなく, 入力設定または出力"L"設定にしておくことを強く推奨します。(ポート方向レジスタの初期設定は入力となっています) MCU がフリーランモードで動作中に BGM アダプタを接続すると, BGM アダプタの 1 ライン UART 端子（ターゲット MCU の DBG/P12 端子）を"L"レベルにする可能性があります(デバッグモードへの切り替え)。このとき,ユーザコード内で DBG/P12 端子が出力"H"に設定されていると, DBG/P12 端子に大電流が流れ, MCU が損傷する恐れがあります。

6 サンプルコード

Start.asm

```
;===============================================================================
; F2MC-8FX Family SOFTUNE C Compiler sample startup routine,
; ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 2008
; LICENSED MATERIAL - PROGRAM PROPERTY OF FUJITSU LIMITED
;===============================================================================
; Sample code for initialization
;-------------------------------------------------------------------------------
        .PROGRAM    start
        .TITLE      start
;-------------------------------------------------------------------------------
; variable define declaration
; #define HWD_DISABLE
; if define this, Hard Watchdog will disable.
; external declaration of symbols
;-------------------------------------------------------------------------------
        .EXPORT     __start
        .IMPORT     _main
        .IMPORT     LMEMTOMEM
        .IMPORT     LMEMCLEAR
        .IMPORT     _RAM_INIT
        .IMPORT     _ROM_INIT
        .IMPORT     _RAM_DIRINIT
        .IMPORT     _ROM_DIRINIT
;-------------------------------------------------------------------------------
; definition to stack area
;-------------------------------------------------------------------------------
        .SECTION    STACK,    STACK,    ALIGN=1
        .RES.B      128-2
STACK_TOP:
        .RES.B      2
;-------------------------------------------------------------------------------
; definition to start address of data, const and code section
;-------------------------------------------------------------------------------
```

```
        .SECTION    DIRDATA,  DIR,      ALIGN=1
        .SECTION    DIRINIT,  DIR,      ALIGN=1
        .SECTION    DATA,     DATA,     ALIGN=1
        .SECTION    INIT,     DATA,     ALIGN=1
;---------------------------------------------------------------------
//code area
;---------------------------------------------------------------------
            .SECTION    CODE,   CODE,   ALIGN=1
__start:
;---------------------------------------------------------------------
; set stack pointer
;---------------------------------------------------------------------
        MOVW    A, #STACK_TOP
        MOVW    SP, A
;---------------------------------------------------------------------
; Set Register bank Pointer 0
;---------------------------------------------------------------------
        MOVW        A,  PS
        MOVW        A,  #0x07FF     // RP=0, DP=0, I=0
        ANDW        A
        MOVW        PS, A
;---------------------------------------------------------------------
; Set ILM to the lowest level (3)
;---------------------------------------------------------------------
        MOVW        A,  PS
        MOVW        A,  #0x0030
        ORW         A
        MOVW        PS, A
;change the main CR clock frequency
        MOV         A , 0x0FE4      ;READ CRTH for NVR trimming value protect
        AND         A , #0x9F       ;Just CLEAR CRTH [6:5]
        OR          A , #0x60       ; Main CR clock update to 8MHz
                                    ;Can update the Main CR clock as below setting
                                    ;0x00: 1MHz;  0x40: 10MHz;    0x60: 8MHz
        MOV 0x0FE4 , A              ;WRITE CRTH to enable the update
;---------------------------------------------------------------------
```

```
; copy initial value *CONST (ROM) section to *INIT(RAM) section
;------------------------------------------------------------------------
#macro   ICOPY          src_addr, dest_addr, src_section
         MOVW           EP, #¥src_addr
         MOVW           A,   #¥dest_addr
         MOVW           A,   #SIZEOF (¥src_section)
         CALL           LMEMTOMEM
#endm

         ICOPY          _ROM_INIT,   _RAM_INIT,   INIT
         ICOPY          _ROM_DIRINIT,    _RAM_DIRINIT,    DIRINIT
;------------------------------------------------------------------------
; zero clear of *VAR section
;------------------------------------------------------------------------
#macro   FILL0          src_section
         MOVW           A,   #¥src_section
         MOVW           A,   #SIZEOF (¥src_section)
         CALL           LMEMCLEAR
#endm
         FILL0          DIRDATA
         FILL0          DATA
;------------------------------------------------------------------------
; call main routine
;------------------------------------------------------------------------
         CALL           _main
end:     JMP            end

;------------------------------------------------------------------------
; Hard Watchdog
;------------------------------------------------------------------------
#ifdef   HWD_DISABLE
         .SECTION       WDT,      CONST,   LOCATE=H'FFBE
         .DATA.W        0xA596
#endif
;------------------------------------------------------------------------
; reset vector
```

```
;------------------------------------------------------------------
        .SECTION    RESET,    CONST,    LOCATE=0xFFFC
        .DATA.B     0xFF
        .DATA.B     0
        .DATA.H     __start

        .END    __start
```

vector.c

```
/* ------------------------------------------------------------------*/
// THIS SAMPLE CODE IS PROVIDED AS IS. FUJITSU MICROELECTRONICS
// ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR OMISSIONS
// Date: 2008.10.09            Version: 1.0
/* ------------------------------------------------------------------*/
// NAME: vector.c
// FUNCTION:      Interrupt level (priority) setting
// Interrupt vector definition
/* ------------------------------------------------------------------*/
#include "mb95200.h"
/* ------------------------------------------------------------------*/
//This function pre-sets all interrupt control registers. It can be
// usedto set all interrupt priorities in static applications. If this
// file contains assignments to dedicated resources, verify that the appropriate
//controller is used.
//-------------------------------------------------------------------
void InitIrqLevels (void)
{
ILR0 = 0xFC;                // IRQ0:   external interrupt ch.4        --> Level 00
                                    // IRQ1:   external interrupt ch.5        --> 01
                                    // IRQ2:   external interrupt ch.2/ch.6  --> 01
                                    // IRQ3:   external interrupt ch.3/ch.7  --> 01
ILR1 = 0xFF;                // IRQ4:   -
                                    // IRQ5:   8/16-bit timer ch.0 (lower)
                                    // IRQ6:   8/16-bit timer ch.0 (upper)
                                    // IRQ7:   LIN-UART (reception)
ILR2 = 0xFF;                // IRQ8:   LIN-UART (transmission)
                                    // IRQ9: -
                                    // IRQ10: -
                                    // IRQ11: -
ILR3 = 0xFF;                // IRQ12:-
                                    // IRQ13: -
                                    // IRQ14: 8/16-bit timer ch1 (upper)
```

```
                                        // IRQ15: -
ILR4 = 0xFF;            // IRQ16:-
                                        // IRQ17: -
                                        // IRQ18: 8/10-bit A/D-converter
                                        // IRQ19: Timebase timer
ILR5 = 0xFF;            // IRQ20: Watch prescaler
                                        // IRQ21:-
                                        // IRQ22: 8/16-bit timer ch1 (lower)
                        // IRQ23: Flash
}
//-----------------------------------------------------------------------
//     Prototypes
// Add your own prototypes here. Each vector definition needed is a
//prototype. Either do it here or include a header file containing them.
//-----------------------------------------------------------------------
//extern unsigned int delay_timer;
__interrupt void DefaultIRQHandler (void);
//-----------------------------------------------------------------------
//Vector definition
//Use the following statements to define vectors.
//All resources related to the vectors are predefined.
//Remaining software interrupts can be added here as well.
//-----------------------------------------------------------------------
#pragma intvect DefaultIRQHandler 0      //   IRQ0:   external interrupt ch0 | ch4
#pragma intvect DefaultIRQHandler 1      //IRQ1: external interrupt ch.5
#pragma intvect DefaultIRQHandler 2      //IRQ2: external interrupt ch.2|ch.6
#pragma intvect DefaultIRQHandler 3      //IRQ3: external interrupt ch.3|ch.7
#pragma intvect DefaultIRQHandler 5      //IRQ5: 8/16-bit timer ch.0 (lower)
#pragma intvect DefaultIRQHandler 6      //IRQ6: 8/16-bit timer ch.0 (upper)
#pragma intvect DefaultIRQHandler 7      //IRQ7: LIN-UART (reception)
#pragma intvect DefaultIRQHandler 8      //IRQ8: LIN-UART (transmission)
#pragma intvect DefaultIRQHandler 14     //IRQ14: 8/16-bit timer ch1 (upper)
#pragma intvect DefaultIRQHandler 18     //IRQ18: 10-biat A/D-converter
#pragma intvect DefaultIRQHandler 19     //IRQ19: Timebase timer
#pragma intvect DefaultIRQHandler 20     //IRQ20:Watch prescaler
#pragma intvect DefaultIRQHandler 22     //IRQ22: 8/16-bit timer ch.1 //(lower)
```

```
#pragma intvect DefaultIRQHandler 23          //IRQ23: Flash
__interrupt void DefaultIRQHandler (void)
{
    __DI();                  // disable interrupts
    while(1)
        __wait_nop();        // halt system
}
```

Main.c

```
/* ---------------------------------------------------------------*/
// THIS SAMPLE CODE IS PROVIDED AS IS. FUJITSU MICROELECTRONICS
// ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR OMISSIONS
// Date: 2008.10.09            Version: 1.0
/* ---------------------------------------------------------------*/
// NAME:           MAIN.C
// FUNCTION:     The following code created for DBG pin usage test
/* ---------------------------------------------------------------*/
#include "mb95200.h"
#define DEBUGMODE
/*------------------------ SUB ROUTINES ---------------------------*/
//set DBG pin as P12 output
void gpio_setting(void)
{
DDR1_P12 = 1;     /* set P12 as output pin */
PDR1_P12 = 0;     /* pull P12 to low. */
}
/*------------------------ Wait loop ------------------------*/
void vDelay (unsigned int uiCount)
{
        while(uiCount--)
        {
                asm ("¥tNOP");
        }
}
/*------------------------ MAIN ROUTINE ----------------------------*/
void main(void)
{
gpio_setting();       //pull low P12 pin

        while(1){
        PDR1_P12 = 0;     /* pull P12 to Llow. */
        vDelay(8000);                   //delay the time
        PDR1_P12 = 1;     /* pull P12 to High */
```

```
        vDelay(8000);           //delay the time

        WDTC=0x35;              //Clear watch dog timer
        }
}
```

Main.c

```
/* ---------------------------------------------------------------*/
// THIS SAMPLE CODE IS PROVIDED AS IS. FUJITSU MICROELECTRONICS
// ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR OMISSIONS
// Date: 2008.10.09              Version: 1.0
/* ---------------------------------------------------------------*/
// NAME:            MAIN.C
// FUNCTION:     The following code created for DBG pin usage test
/* ---------------------------------------------------------------*/
#include "mb95200.h"
#define DEBUGMODE       // Input port change
/*------------------------ SUB ROUTINES ---------------------------*/
//set DBG pin as P12 output
void gpio_setting(void)
{
DDR1_P12 = 0;     /* set P12 as input pin */
AIDRL_P04 = 1;    /* port input enable */
DDR6_P63 = 1;     /* set P63 as output pin */
PDR6_P63 = 0;     /* pull P63 to low. */
}

//set DBG pin as EC0 pin
void p12_as_ec0(void)
{
SYSC = 0xE3;       // set P12 as EC0 pin, disable HCLK1
T00CR0 = 0x71;    // interval timer, external clock source.
TMCR0 = 0x43;      // 8bit mode, output timer00.
T00DR = 0x08;     // set counter.
T00CR1 = 0x81;    // enable output, start timer00.
}
//change EC0 pin to P04
void p04_as_ec0(void)
{
SYSC = 0xEB;       // set P04 as EC0 pin, disable HCLK1
T00CR0 = 0x71;    // interval timer, external clock source.
```

```
TMCR0 = 0x43;     // 8bit mode, output timer00.
T00DR = 0x08;     // set counter.
T00CR1 = 0x81;    // enable output, start timer00.
}

/*------------------------ Wait loop --------------------------*/
void vDelay (unsigned int uiCount)
{
            while(uiCount--)
            {
                        asm ("¥tNOP");
            }
}
/*------------------------ MAIN ROUTINE ----------------------------*/
void main(void)
{
gpio_setting();       //pull low P12 pin

#ifdef DEBUGMODE
p04_as_ec0();          //set P04 as EC0 pin in debug mode
#else
p12_as_ec0();          //set P12 as EC0 pin in free run mode
#endif
            while(1){
            PDR6_P63 = 0;     /* pull P63 to Llow. */
            vDelay(1000);                  //delay the time
            PDR6_P63 = 1;     /* pull P63 to High */
            WDTC=0x35;                     //Clear watch dog timer
            }
}
```